GIS Topology

An ESRI® White Paper • July 2005

ESRI 380 New York St., Redlands, CA 92373-8100, USA • TEL 909-793-2853 • FAX 909-793-5953 • E-MAIL info@esri.com • WEB www.esri.com

GIS Topology

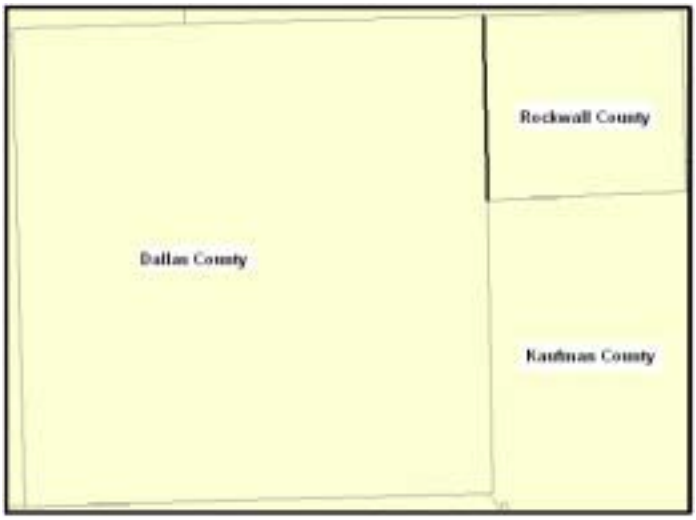
An ESRI White Paper

Contents **Page**

GIS Topology

Topology

A GIS topology is a set of rules and behaviors that model how points, lines, and polygons share geometry. For example, adjacent features, such as two counties, will share a common edge.

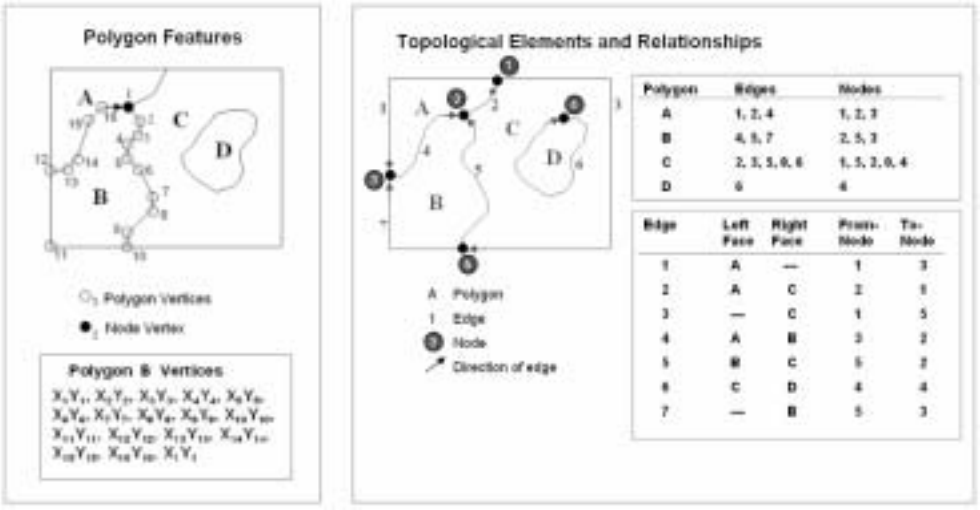


Shared Boundary between Dallas and Rockwall Counties in Texas

Further geometrical relationships might be defined, for example, county boundaries must completely cover states and share edges along state boundaries. These are examples of topological rules and behaviors that are commonly used in GIS databases.

Topology has long been a key GIS requirement for data management and integrity. In general, a topological data model represents spatial objects (point, line, and area features) using an underlying graph of topological primitives. These primitives, together with their relationships to one another and to the features whose boundaries they represent, are defined by representing the feature geometries in a planar graph of topological elements. Such datasets are said to be topologically integrated.

J-9469



This illustration shows how a layer of polygons can be described in two ways: (1) collections of geometric features and (2) a graph of topological elements (nodes, edges, faces, and their relationships).

This means that there are two potential methods used when working with features—one in which features are defined by their coordinates and another in which features are represented as an ordered graph of their topological elements.

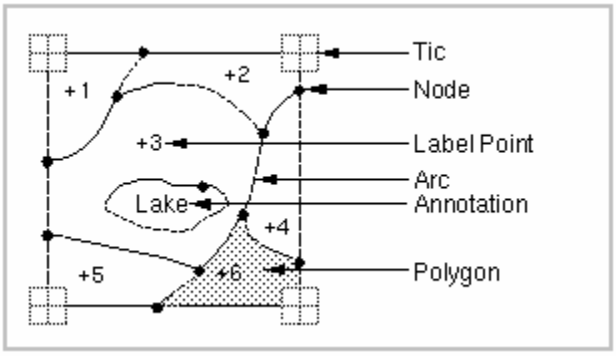
Why Topology?

Topology is employed to

- Manage shared geometry (i.e., constrain how features share geometry). For example, adjacent polygons, such as parcels, share edges; street centerlines and the boundaries of census blocks share geometry; adjacent soil polygons share edges.
- Define and enforce data integrity rules (e.g., no gaps should exist between parcel features, parcels should not overlap, road centerlines should connect at the endpoints).
- Support topological relationship queries and navigation (e.g., have the ability to identify adjacent and connected features, find the shared edges, and navigate along a series of connected edges).
- Support sophisticated editing tools that enforce the topological constraints of the data model (e.g., ability to edit a shared edge and update all the features that share the common edge).
- Construct features from unstructured geometry (e.g., the ability to construct polygons from lines sometimes referred to as "spaghetti").

Historical Topological Data Model Example: The ArcInfo Coverage

ArcInfo® coverage users have a long history and appreciation for the role that topology plays in maintaining the spatial integrity of their data.



Elements of the ArcInfo Coverage Data Model

In a coverage, the feature boundaries and points were stored in a few main files that were managed and owned by ArcInfo Workstation. The ARC file held the linear or polygon boundary geometry as topological edges, which were referred to as "arcs." The LAB file held point locations, which were used as label points for polygons or as point features such as a set of points representing oil well locations. Other files were used to define and persist the topological relationships between each of the edges and polygons. For example, one file called the Polygon-arc list (PAL) file listed the order and direction of the arcs in each polygon. In ArcInfo, software logic was used to assemble the coordinates for each polygon for display, analysis, and query operations. The ordered list of edges in the PAL file was used to look up and assemble the edge coordinates held in the ARC file. The polygons were assembled during runtime when needed.

The coverage model had numerous advantages:

- It used a simple structure to maintain topology.
- It enabled edges to be digitized and stored only once and shared by many features.
- It could represent polygons of enormous size (with thousands of coordinates) because polygons were really defined as an ordered set of edges (or arcs).
- The topology storage structure of the coverage was intuitive. Its physical topological files were readily understood by ArcInfo users.

Coverages also had some disadvantages:

- Some operations were slow because many features had to be assembled on the fly when they needed to be used. This included all polygons and multipart features such as regions and routes.
- Topological features (such as polygons, regions, and multipart lines called "routes") were not ready to use until the coverage topology was built. If edges were edited,

the topology had to be rebuilt. (Note: Partial processing was eventually used, which required rebuilding only the changed portions of the coverage topology.) In general, when edits are made to features in a topological dataset, a geometric analysis algorithm must be executed to rebuild the topological relationships regardless of the storage model.

- Coverages were limited to single-user editing. Because of the need to ensure that the topological graph was synchronized with the feature geometries, only a single user at a time could update a topology. Users would tile their coverages and maintain a tiled database for editing. This enabled individual users to "lock down" and edit one tile at a time. For general data use and deployment, users would append copies of their tiles to a mosaicked data layer.

Shapefiles and Simple Geometry Storage

In the early 1980s, coverages were seen as a major improvement over the older polygon and line-based systems in which polygons were held as complete loops. In these older systems, all the coordinates for a feature were stored in each feature's geometry. Before the coverage and ArcInfo came along, these simple polygon and line structures were used. These data structures were simple but had the disadvantage of double-digitized boundaries. That is, two copies of the coordinates of adjacent portions of polygons with shared edges would be contained in each polygon's geometry. The main disadvantage was that GIS software of that time could not maintain shared-edge integrity. In addition, storage costs were extremely high, and each byte of storage came at a premium. During the early 1980s, a 300 MB disk drive was the size of a washing machine and cost $30,000. Holding two or more representations of coordinates was expensive, and computations took too much compute time. Thus, the use of a coverage topology had real advantages.

During the mid-1990s, interest in simple geometric structures grew because disk storage and hardware costs in general were decreasing while computational speed was increasing. At the same time, existing GIS datasets were more readily available, and the work of GIS users was evolving from primarily data compilation activities to include data use, analysis, and sharing.

Users wanted faster performance for data use (for example, do not spend computer time to derive polygon geometries when we need them. Just deliver the feature coordinates of these 1,200 polygons as fast as possible). Having the full-feature geometry readily available was more efficient. Thousands of GIS systems were in use, and numerous datasets were readily available.

Around this time, ESRI had developed and published its ESRI® shapefile format. Shapefiles used a very simple storage model for feature coordinates. Each shapefile represented a single feature class (of points, lines, or polygons) and used a simple storage model for the feature's coordinates.

A few years later, ArcSDE® pioneered a similar simple storage model in relational databases. A feature table could hold one feature per row with the geometry in one of its columns along with other feature attribute columns.

FID	Shape	AREA	PERIMETER	STATES07	STATES_ID	STATE_FIPS	STATE_NAME
39	Polygon	106889454880	1587963.5	39	81	39	Ohio
40	Polygon	49636880	96674.375	40	74	36	New York
41	Polygon	145811518840	1970546.5	41	58	17	Illinois
42	Polygon	124800884	4761351338125	42	75	36	New York
43	Polygon	94274002880	1608101.625	43	88	18	Indiana
44	Polygon	269618129409	2150314.75	44	73	8	Colorado
45	Polygon	62781031286	1750380.125	45	76	54	West Virginia
46	Polygon	5331179432	416392.40875	46	78	10	Delaware
47	Polygon	25874441884	1008772.25	47	80	24	Maryland
48	Polygon	101817407382	2642379.75	48	81	51	Virginia
49	Polygon	180883706350	2289698.5	49	77	29	Missouri
50	Polygon	86528333	74416.53125	50	83	24	Maryland
51	Polygon	171103440	62137.86546875	51	84	11	District of Columbia
52	Polygon	13748660	20886.21836875	52	85	24	Maryland
53	Polygon	212888044872	1974101.125	53	78	20	Kansas
54	Polygon	46307840	74808.619625	54	86	24	Maryland
55	Polygon	33503928	49524.83125	55	87	51	Virginia
56	Polygon	1481258380	287998.28125	56	88	51	Virginia
57	Polygon	104374643744	1985348.75	57	82	21	Kentucky
58	Polygon	294512688912	2334578	58	89	4	Arizona
59	Polygon	181248820	187588.828125	59	90	37	North Carolina
60	Polygon	51395740	33584.08840525	60	91	37	North Carolina
61	Polygon	12888881376	2604036.25	61	93	37	North Carolina
62	Polygon	315348600130	2379620.75	62	81	35	New Mexico
63	Polygon	127220980	161900.3125	63	93	37	North Carolina
64	Polygon	109081418672	1997350.875	64	82	47	Tennessee
65	Polygon	181388089380	2485936.25	65	93	40	Oklahoma
66	Polygon	41696843	54628.21875	66	101	37	North Carolina
67	Polygon	55621894	26955.25	67	94	21	Kentucky
68	Polygon	686808022912	5960369	68	97	48	Texas

Example of a feature table of state polygons for the United States. Each row represents a state's geographic shape and its attribute properties. The Shape column holds the polygon geometry.

This "simple features" model fit the SQL processing engine well. Through the use of relational databases, we began to see GIS data scaled to unprecedented sizes and numbers of users without degrading performance. We were beginning to leverage RDBMS for GIS data management.

Shapefiles became ubiquitous and, using ArcSDE, this simple features mechanism became the fundamental feature storage model in RDBMS systems. (Eventually, ESRI became the lead author of the Open Geospatial Consortium, Inc., and ISO simple features specification because we recognized the significance of this simple feature storage mechanism.)

It had clear advantages:

- The complete geometry for each feature was held in one record. No assembly was required.
- The data structure (physical schema) was simple, fast, and scalable.
- It was easy for programmers to write interfaces.
- It was interoperable. Many wrote simple converters to move data in and out of these simple geometries from numerous other formats. Shapefiles were widely applied as a data use and interchange format.

Its disadvantages were that maintaining the data integrity readily provided by topology was not as easy to implement for simple features. As a consequence, users applied one data model for editing and maintenance (such as coverages) and used another for deployment (such as shapefiles or ArcSDE layers).

Users began to use this hybrid approach for editing and data deployment. For example, users would edit their data in coverages, CAD files, and so forth. Then, they would convert their data into shapefiles for deployment and use. Thus, even though the simple features structure was an excellent direct use format, it did not support the topological editing and data management of shared geometry. Direct use databases would use the simple structures, but another topological form was used for editing. This had advantages for deployment. However, the disadvantage was that data would become out of date and have to be refreshed. It worked, but there was a lag time for information update. Bottom line—topology was missing.

What GIS required and what the geodatabase topology model implements now is a mechanism that stores features using the simple feature geometry but enables topologies to be used on this simple, open data structure. This means that users can have the best of both worlds—not only a transactional data model that enables topological query, shared geometry editing, rich data modeling, and data integrity but also a simple, highly scalable data storage mechanism that is based on open, simple feature geometry.

This direct use data model is fast, simple, and efficient. It can also be directly edited and maintained by any number of simultaneous users. With the release of ArcGIS® 8.3 and beyond, users began to discover the advantages of this new topology implementation.

Research Results for Topology Management in ArcGIS

The user requirements for topology support in any GIS are well known. These include

- Support working with both the feature coordinates and their underlying topological graph including the ability to work simultaneously with both representations.
- Support large datasets (up to millions of features) with large features (millions of coordinates per feature).
- Allow many simultaneous editors and updates.
- Support common topological operations (e.g., traverse the connected edges in a topology, find adjacent features, update a shared edge between features, vertically integrate features such as states and counties).
- Support GIS workflows involving long transactions and multiple versions.

These functions need to be supported regardless of the underlying data storage model. What is critical is that these capabilities must be flexibly supported by the storage mechanism used.

At ESRI, we investigated a number of alternate implementations of the logical topology model and software that would support this functionality. We knew the coverage limitations. Eventually, users would hit limits on the number of features in a dataset.

Data maintenance was limited to single-user updates to the topological graph. The transaction model was closed, and so on.

We also had a strong conviction (along with the rest of the software industry) that the simple features storage model held great scalability and transaction support and fully leveraged the scalability provided by relational database technology. Simple feature storage held some very clear advantages.

In our development efforts, we began to discover that a kind of reverse process was possible during shapefile editing. Features could be decomposed into their topological elements of edges, faces, and nodes as needed. For example, a user could "discover" and edit the shared edges between two adjacent polygons.

In other words, asking the GIS software for topological elements (such as nodes, edges, and faces) when needed was as fast or faster than assembling polygon and line features from the underlying coverage's topology graph. Users could also efficiently traverse the topological graph of elements using software logic (find connected edges, locate adjacent features, update the shared edges, etc.).

This meant that features could be stored in tables using simple geometries. Simple feature coordinates could be topologically analyzed and indexed to identify shared geometries between features. In addition, topological graphs could be served by GIS software logic from the simple features storage model.

This requires a few key ingredients such as

- A persisted topological index on the coordinates of the simple features.
- Advanced software logic that knows how to analyze and discover the topological elements in the feature geometry.
- An editing and data management framework that can maintain topological integrity.
- A data management framework that includes the ability to define the integrity rules and topological behavior of feature classes.
- A rich set of tools to validate, discover, identify, edit, and resolve both the topological graph and the feature coordinates.
- GIS software logic that can navigate topological relationships, work with adjacency and connectivity, and assemble features from these elements. For example, show me the polygons that share a specific common edge. Give me a list of the edges that connect at a certain node. Navigate along connected edges from the current location. Add a new line and "burn" it in to the topological graph; split it at intersections; and create resulting edges, faces, and nodes.
- The ability to deploy the topological software logic on desktops, in custom applications, on server frameworks, and on the Web.

In addition, the software and database schemas had to scale to support enormous data volumes and many simultaneous editors.

The Topology Solution in ArcGIS

In effect, topology has been considered as more than a data storage problem. It needs to be considered as part of a complete solution. This solution includes

- A complete data model (objects, integrity rules, editing and validation tools, a topology and geometry engine that can process datasets of any size and complexity, and a rich set of topological operators and query tools)
- An open storage format using a set of record types for simple features and a topological index on this format
- The ability to work with features (points, lines, and polygons) as well as topological elements (nodes, edges, and faces) and their relationships to one another

This also must include a mechanism that would support

- Massively large datasets with millions of features
- The ability to perform editing and maintenance by many simultaneous editors
- Always ready-to-use, always available feature geometry
- Always ready-to-use, always available topological elements
- Topological integrity and behavior
- A system that goes fast and scales for many users and many editors
- A system that is flexible and simple
- A system that leverages the RDBMS SQL engine and transaction framework
- A system that can support multiple editors, long transactions, historical archiving, and replication

After testing numerous implementations, we made the pragmatic decision to use simple geometry storage for all features in the geodatabase and to index the shared coordinates on each feature using a specialized topological index. In ArcSDE and the geodatabase, we already had experience using indexes and software logic to model advanced behaviors on the simple feature storage model. We had witnessed firsthand the clear advantages of direct use data formats and their ability to scale and support many simultaneous users.

In the case of topology, we implemented a specialized coordinate index in which shared coordinates between features (both in the same feature class and across feature classes) are the same in all feature geometry. Each of these shared coordinates also shares a common identifier. A topological index of these coordinates allowed us to navigate topological relationships between features that share coordinates and their topological elements. These coordinates are stored as part of each feature's simple geometry, and

topological indexes on the coordinates enable each shared coordinate to be used as foreign keys to other features. This enables fast and scalable lookup of topological elements (nodes, edges, and faces). In other words, we can readily define and use relationships between shared coordinates for performing topological queries and updates. This has the added advantage of working quite well and scaling with the RDBMS' SQL engine and transaction management framework.

As features are added during editing and update, they are directly usable. At any time, users can choose to topologically analyze and validate updated areas, which are tracked as updates are made to each feature class.

The results are that topological primitives (nodes, edges, and faces) and their relationships to one another and their features can be efficiently discovered and assembled. As mentioned previously in the coverage, we assembled the feature geometry on the fly from the topological elements when it was needed. In this case, we reverse the process and retrieve the topological primitives (i.e., the nodes, faces, and edges) on the fly when they are needed along with their relationships.

This has several advantages:

- Simple feature geometry storage is used, which is open, efficient, and scales to large sizes and numbers of users.

- This physical data model is transactional and multiuser. By contrast, the older topological storage models will not scale. Supporting multiple editor transactions and numerous other GIS data management workflows is extremely difficult. It also means that geodatabase topologies need not be tiled, many users can update the topological database simultaneously, and a data layer can grow to any size (e.g., hundreds of millions of features) with strong performance.

- This topology implementation is additive. Users can typically add this to an existing schema of spatially related feature classes instead of redefining and converting all of their existing feature classes to new data schemas containing topological primitives.

- There need only be one data model for geometry editing and data use, not two or more.

- It is interoperable because all feature geometry storage adheres to simple features specifications from the Open Geospatial Consortium and ISO.

Data modeling is more natural because it is based on user features (such as parcels, streets, soil types, and watersheds) instead of topological primitives (such as nodes, edges, and faces). Users will begin to think about the integrity rules and behavior of their actual features instead of the integrity rules of the topological primitives. For example, how do parcels behave? This will enable stronger modeling for all kinds of geographic features. It will improve our thinking about streets, soil types, census units, watersheds, rail systems, geology, forest stands, landforms, physical features, and so on.

It provides the same information content as persisted topological implementations—either users store a topological line graph and discover the feature geometry or they store the feature geometry and discover the topological elements and relationships.

In cases in which users want to store the topological primitives, it is easy to create and post topologies and their relationships to tables for various analytical and interoperability purposes (such as users who wish to post their features into an Oracle Spatial warehouse that stores tables of topological primitives).

At a pragmatic level, the ArcGIS topology implementation works. It scales to extremely large geodatabases and multiuser systems without loss of performance. It includes rich validation and editing tools for building and maintaining topologies in geodatabases. It includes rich and flexible data modeling tools that enable users to assemble practical, working systems on any relational database and any number of schemas.